Summary of Project 1

Design Ideas:

1. Task 1
   1. No design ideas, just setting up environment
2. Task 2
   1. No design ideas, just copying code
3. Task 3
   1. Use the getopt function to parse the arguments like how the example used a, b, and c
   2. When the -v flag is there, print the version using values from lab.h
   3. Exit the program after printing, don’t run anything else and no need to allocate any memory outside of this function
4. Task 4
   1. Use readline and history from the GNU library
   2. Loop to keep asking for input
   3. Store the input into history
   4. Free input because of memory leaks
5. Task 5
   1. Use getenv() to retrieve the MY\_PROMPT environment variable
   2. If the variable is set, use its value as the prompt; otherwise, fall back to a default prompt: ‘shell> ‘
6. Task 6
   1. Make all of the builtin cmds in dobuiltin, use if statements to check if its:
      1. exit: Exit the shell and clean up
      2. cd: Change the directory using chdir(), and if there are no arguments then change to home
      3. history: Print the command history using print\_history() with the using\_history stuff and adding every item to the history with add\_history()
7. Task 7
   1. Pass in args for execvp to run command and get the args for it
   2. Fork a new process
   3. Use waitpid() in the parent to wait for the process to finish
   4. If pid is 0 then it is child
8. Task 8
   1. Use signal function to ignore things like SIGINT, SIGQUIT, SIGTSTP in the shell
   2. Make sure they are back to default before running process
9. Task 9
   1. Look for & to see if it should be in the background
   2. Print job statuses on enter
   3. Track jobs with an array if background
10. Task 10
    1. Add jobs command to dobuiltin
    2. Print all jobs with jobs list and statuses

Functions I achieved with my Shell:

1. First function I was able to implement was version printing in task 3. When you type ./myprogram -v, it will print the version and exit the shell
2. In task 5, I made the ability to change the prompt with the MY\_PROMPT environment variable. The default prompt will be ‘shell> ’
3. In task 6, I implemented the exit cmd, that will exit the shell and clean up memory
4. In task 6, I implemented the cd cmd, which will switch directories with chdir(). If there is an argument of a directory, it will take you to take directory, and if there is no argument it will take you to the home directory
5. In task 6, I implemented the history cmd, which will utilize the using\_history() function and the fact that I have been adding all user input to the add\_history() function to print all of the history in the past for the shell.
6. In task 7, I used the execvp() command to run basically any other command such as ls, pwd, date, echo, etc.
7. I made it so signals that will terminate or suspend the shell are ignored except for when commands are running. This was in task 8.
8. In task 9, if you add the ‘&’ operator, it will have the cmd run in the background. You can get updates by pressing enter.
9. In task 10, I implemented the jobs cmd, so you can see current jobs and completed jobs. Once jobs is called, all previous jobs with ‘Done’ status are removed.